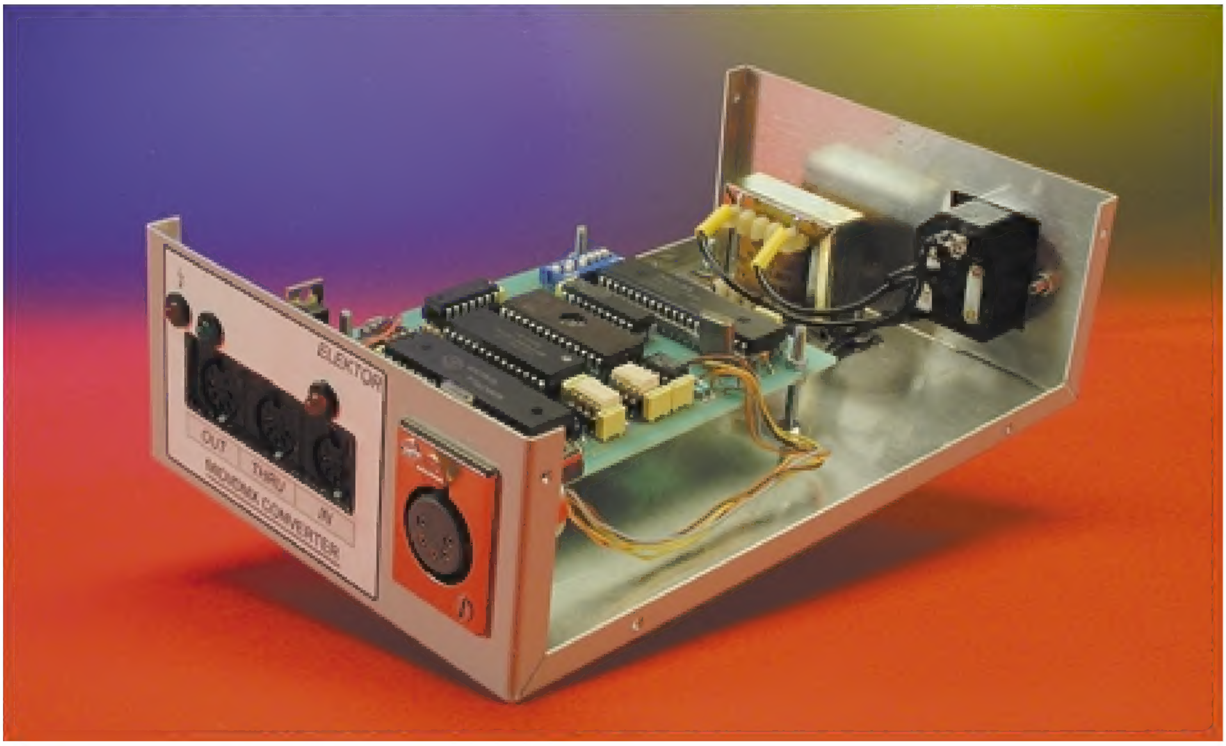
MIDI-DMX Interface

control your DMX network from a sequencer !

Design by B. Bouchez

We've recently described the principle of DMX512-based remote control, which allows automated projectors or flood lights to be controlled on up to 512 channels, via an RS485 link. Every DMX system comprises a master and several slave devices responding to DMX commands.



Actually, there are two approaches for the realisation of a DMX512 master: either a dedicated controller is used (such as the MARTIN2518, for example), or software running on a PC.

The main advantage of the first approach lies in its simplicity and low cost. On the down side, the capabilities, especially in programming and memory, are generally limited. Moreover, the user interface of some of these controllers is sometimes original if not odd, to put it mildly.

As always, the use of a PC with a DMX interface allows you to do almost everything, while offering unbeatable capabilities. Actually, there is almost no concert, public event or TV show without some form of PC-controlled (stage) lighting.

However, the PC solution also has two disadvantages: first, the need of

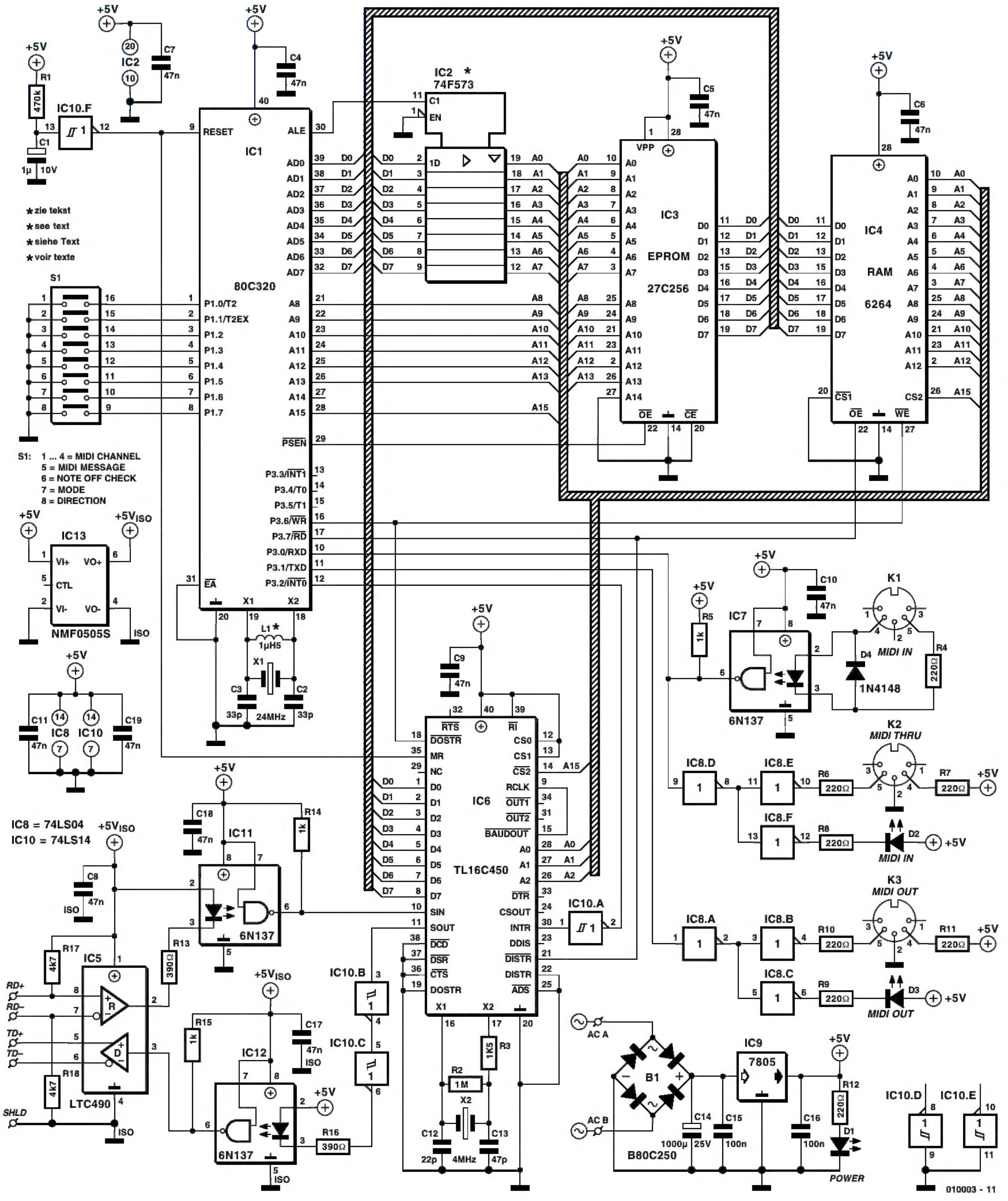


Figure 1. This circuit diagram of the DMX-to-MIDI interface effectively shows a stand-alone microcontroller system.

a specific interface, since the PC's serial port is not compatible with the DMX512 standard; second, suitable software to support these interfaces is costly an difficult to use for non-specialists.

The interface described here allows, at a small outlay, to control any DMX system from any MIDI sequencer, on PC, Macintosh or stand-alone. Rather than controlling sounds, the sequencer controls DMX levels, colour changes, gobo calls, etc.

This interface integrates another very useful function: a DMX to MIDI converter that enables you to look what happens on your DMX control and identify the main problems that can arise. Moreover, this function can be used to capture sequences generated by DMX controller in your own sequence software.

Lastly, a suite of software utilities has been designed especially for this interface, including a driver library for easy creation of your own software under Windows. This toolkit also includes two test programs.

Finally, the author of this project has written two powerful but easy to use light control programs employing the interface described here.

The circuit

As you could imagine, the interface is built around a microcontroller in charge of various dataflows. Initially, the well-known 80C32 was chosen, but following a field test of the prototype it transpired that the software load on the CPU was rather too high for reliable operation. In the end the 80C320 was elected, which, being pin compatible as well as software compatible with the 80C32, runs three to four times faster thanks to a completely redesigned kernel. None the less, as we will see a bit further on, it is possible to use a 80C32 in a 'light' version of the interface. To support the fast data transfers (250 kbits/s on DMX, 31.25 kbits/s on MIDI), the micro-controller is clocked at 24 MHz.

Although the current version of the software works perfectly, it should be able to improve or extended it in the future. Consequently a external EPROM (IC3) is used to hold the program. Don't be surprised to see an address line tied to ground rather than to the 80C320 address bus. Although the software may also be loaded into a 27C64 or 27C128, these EPROMs are generally slower and more costly than the 27C256. Because not all memory is actually used, unused pins have been tied to ground to simplify the PCB drawing. Please note that the EPROM speed must be 120 ns or less, due to the processor speed. For this same reason, the data/address demultiplexer, IC2, should normally be a 74**F**573. The use of a 74HCT573 is only required in a few rare cases, see the section on testing further on.

Because the amount of data to be handled exceeds the processor's on-chip memory capacity, an external RAM is used (IC4). As will be explained a bit further on, it is possible to omit this memory device at the cost of some interface capabilities. As for with EPROM, a smaller RAM device could be used, but these 'small' memories may be hard to find, especially devices with a fast access time. A relatively large memory is used here, which should be easy to obtain at a small additional cost. Again, it is mandatory to use a 120 ns or faster memory.

The MIDI interface is built directly around the microcontroller, using its internal serial port. The physical interface around IC7 and IC8 is classical and fully complies with the MIDI specification. Two signalling LEDs were added to visualize the data flow on MIDI IN and MIDI OUT. This kind of extension costs next to nothing but is really practical in actual use when problems arise between the interface and the sequencer.

At the DMX side, the interface is built around IC6, a Type 16C450 fast UART (Universal Asynchronous Receiver Transmitter). This chip is the evolution of the 'good old' 8250, which has been part and parcel of the PC's serial port since the beginning. The 16C450 and 16C550 can easily achieve data throughput in excess of 250 kbits/s, where the 8250 is limited to 19,200 bits/s (even it can reach 115.2 kbits/s in practice).

This UART is a 'must have' in this circuit because it greatly simplifies the job of the microcontroller. The UART circuit integrates a Break detector, which is used for DMX synchronisation. On the 80C320's serial port, such detection would have to be implemented in software.

The UART integrates its own crystal oscillator and a programmable divider that allows almost any baud rate to be set. Note that a clock signal, which is 16?, the baudrate may be found on the RCLK pin of the chip. On this board, you can find a 4-MHz (16?250 kbits/) frequency. This is very practical when testing the board.

You will notice that IC10 is a 7414 Schmitt trigger. This IC is used to create the reset pulse for the UART and microcontroller, which must start simultaneously for software reasons. A 7404, which has the same pinout, but no specified trigger behaviour, can lead to serious problems when the board starts up. For the record, the author lost several days on one of the prototypes chasing a non-existing software bug, only to discover that the UART started after the micro-controller.

An LTC490 converter with separate send and receive buffers is employed to transform TTL levels (used by UART) into RS485 levels. Resistors R17 and R18 are biasing devices on the DMX lines and serve to avoid false 'Break' detections when a DMX input is not connected. Without these resistors, a false Break condition would cause erroneous synchronisation.

Especially in long distance DMX networks, there is a real risk of earth loops being created. To prevent problems, the DMX side is optically isolated from the rest of the circuit. Since DMX is quite fast, it is mandatory to use high speed 6N137 opto-couplers.

This part of the circuit is powered by a small DC/DC converter. It is possible to lower the cost of the interface by omitting the converter and the opto-couplers. In that case, you have to connect the SOUT pin of the UART directly to DIN (pin 3) input of LTC490, and the SIN pin of the UART to DOUT (pin 2) of the LTC490. Remember, however, that this configuration greatly increases the risk of damage to the interface in case of energy return on DMX512 lines. From our experience, it happens more frequently that you could think.

Last thing, the interface is configured by an octal (8-way) DIP switch, S1, directly connected to the processor's P1 port. There is no need for pull-up resistors, since these are integrated in the P1 port.

The software

The value of a microcontroller-driven board is totally dependent on the software it hosts. In the case of the present interface, a lot of software is involved! To give you an idea, DMX-receive processing can use up to 60% of processor's computing capabilities. On the MIDI side, it can reach 45% for a heavily loaded (= very busy) MIDI line. Moreover, receiving data is only a part of the functions carried out by the CPU, transmission in the background being the other.

Before detailing the different modes of the interface, let's take a look to some generic things about it.

On the MIDI side, our interface recognises two message types: Note On (90h) and Control Change (B0h). The message type used for the MIDI-to-DMX as well as the DMX-to-MIDI direction is selected by means of switch **S1-5**. With the switch in ON position, the interface employs Control Change messages. In the other case (OFF position), the interface uses Note On messages.

These two MIDI messages were chosen because they comprise two data bytes, the first for note or control number, the second for the value of velocity or control.

Since a MIDI channel can transmit 128 notes or controls, the first byte will be used to identify the corresponding DMX channel.

The second byte of data will be used to set the DMX value for the channel selected by the previous byte. As MIDI supports only 128 values (DMX supporting 256), a scaling operation is performed by truncating the least-significant bit and subsequent rotation. The correlation between MIDI and DMX values is

Table 1. MIDI/DMX values correspondence

MIDI Value	◀▶	DMX Value
0		0
0		1
1		2
1		3
2		4
2		5
3		6
3		7
.		.
.		.
126		252
126		253
127		254
127		255

Table 2. Mode selection by S1-7 and S1-8

S1-7 (Basic Mode/ 4 channels)	S1-8 (Direction)	Mode
OFF	OFF	1
OFF	ON	2
ON	OFF	3
ON	ON	4

given in **Table 1**.

Switch **S1-6** allows or disallows recognition of Note Off messages (to make sense to this switch, the interface should be in Note On mode, not Control Change). If S1-6 is closed, the interface will treat any Note Off received on configured MIDI channel(s) as a blackout command for the corresponding DMX channel. Thanks to this function, programming MIDI sequencers is generally easier. In Modes 2 and 4, S1-6 has a different function, allowing a special mode to be activated, called 'Delta'. In Delta Mode, only those MIDI values that have changed since the previous cycle are returned to the MIDI. Obviously, this considerably lightens the traffic on the MIDI link. More importantly though, it allows DMX sequences to be registered on a MIDI sequencer.

Table 4. Selection of MIDI channel by S1-3 and S1-4 for Modes 3 and 4

S1-3	S1-4	MIDI Channels
OFF	OFF	1, 2, 3, 4
ON	OFF	5, 6, 7, 8
OFF	ON	9, 10, 11, 12
ON	ON	13, 14, 15, 16

The protocol converter of the interface can be used under four modes, selected by **S1-7** and **S1-8**, see **Table 2.**

Mode 1: the converter transforms MIDI data on the channel selected by S1-1 to S1-4 (see table 3) into 128 DMX values.

Mode 2: the converter transforms the first 128 DMX channels received into MIDI messages and send them on MIDI OUT to the MIDI channel selected by S1-1 to S1-4. MIDI message content is defined by S1-5 (see below). Note that MIDI transmission is continuous to insure that data are regularly refreshed.

Mode 3: the converter controls 512 DMX channels on the DMX output (this is called a 'DMX world'). Since it is not possible to send 512 values on a single MIDI channel, the interface will use 4 consecutive MIDI channels, each of these controlling 128 values. In this mode, only S1-3 and S1-4 give the base MIDI channel, the three others being the following channels. Possible values for the MIDI set-up are given in **Table 4**.

Mode 4: in this mode, the converter transforms the 512 received DMX channels into MIDI messages and sends them continuously to MIDI OUT. As in previous mode, four consecutive MIDI channels will be used, selected by S1-3 and S1-4 (see Table 4). As with Mode 2, the MIDI message type is selected by switch S1-5.

Note that, in all cases, the interface recognises and transmits DMX frames with 'Start Code' equal to 0 only.

Building the interface

Although the schematics may appear quite complex, building this interface is relatively simple, mainly thanks to the double-sided PCB specially designed for the project. As always, it is highly recommended to use high quality sockets for all ICs, especially the microcontroller and its peripheral devices.

Table 3. MIDI Channel selection by S1-1 through S1-4 for Modes 1 and 2

S1-1	S1-2	S1-3	S1-4	MIDI channel
OFF	OFF	OFF	OFF	1
ON	OFF	OFF	OFF	2
OFF	ON	OFF	OFF	3
ON	ON	OFF	OFF	4
OFF	OFF	ON	OFF	5
ON	OFF	ON	OFF	6
OFF	ON	ON	OFF	7
ON	ON	ON	OFF	8
OFF	OFF	OFF	ON	9
ON	OFF	OFF	ON	10
OFF	ON	OFF	ON	11
ON	ON	OFF	ON	12
OFF	OFF	ON	ON	13
ON	OFF	ON	ON	14
OFF	ON	ON	ON	15
ON	ON	ON	ON	16

As already mentioned, it is possible to build a 'light' version of the interface. In that case, you can only use Mode 1 (which is the most common and useful mode). For this version, there is no need for SRAM IC4, which consequently need not mounted. Since DMX-receive is inactive in Mode 1, there is no need to mount IC11, R13 and C18. Last but not least, it is possible to use the good old 80C32, which will be cheaper than the 80C320.

While on the subject of these controllers, some retailers are unable to specify the maximum frequency of the ICs they sell. Worse, they will not hesitate to sell 16 or 20 MHz devices processors and claim compatibility with a 24MHz clock. We even surprised one day a retailer supplying an 8052 (which is NMOS and limited to 12 MHz) as a 80C32 (CMOS), saying that it will only run a little warmer to the touch. You have to know that a processor specified at 20 MHz may seem to work correctly at 24MHz, but will cause problems one day. We expericed the case of a 80C32 whose serial port was totally 'crazy' during interrupt processing, when the rest of the chip was running correctly.

For this project, if you want to build the 'light' version, we recommend to use Philips 80C32 -IBPN, -UBPN or -IFPN versions, which are guaranteed at 24 MHz. As far as we know, Intel processors of this family are limited to 20 MHz, but that may nedd to be verified. If you want to make the 'full' version, you have to use a Dallas Semiconductor 80C320MCG (this is the most frequently seen version).

For the UART, you can use the Texas Instruments TL16C450 or National Semiconductor NS16C450. You can also use TL16C550

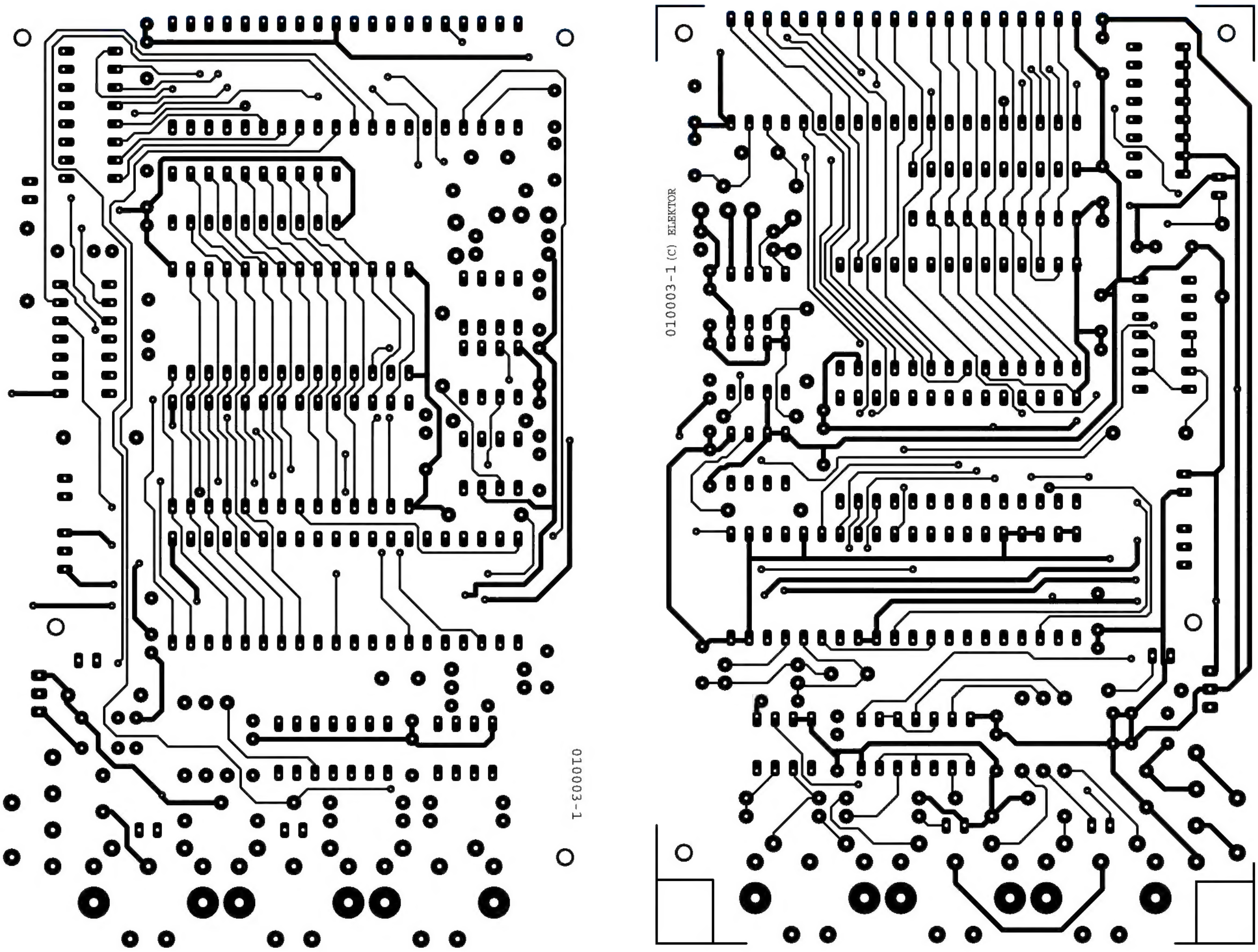


Figure 2. Copper track layout and component mounting plan of the PCB designed for the interface (board available ready-made).

or equivalent without any problem. These components may be found with more generic type numbers like 16450, 16C450, 16550 or 16C550. Note that you may find such components on serial port add-on boards for PCs (such boards may be picked up cheaper than the component alone). Do not use 82C50 or 8250, since these chips are normally not compatible with 4-MHz crystals.

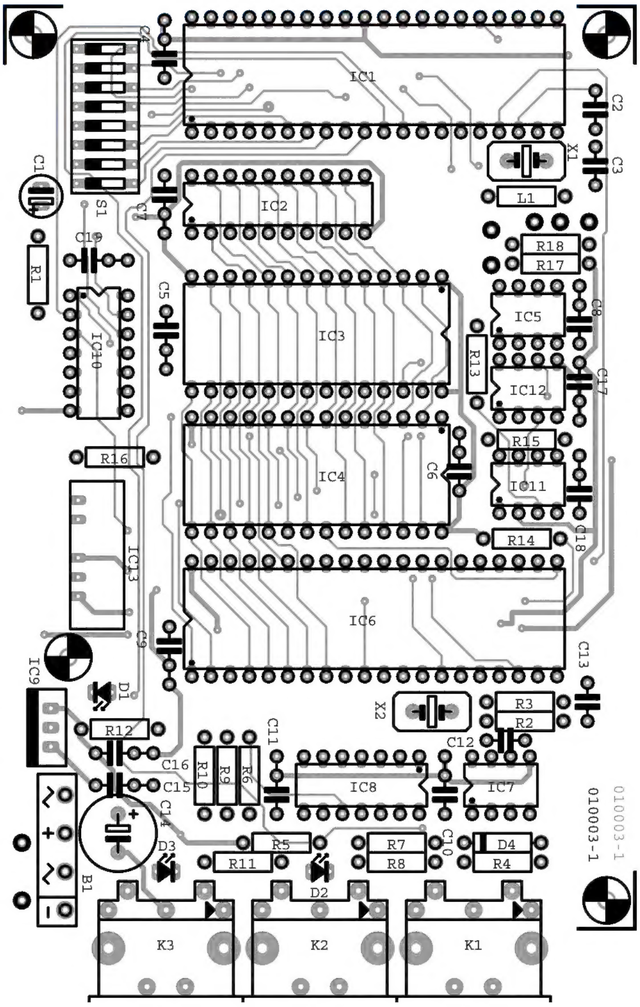
The only problematic component (as far as availability is concerned) is the DC/DC converter IC13, a NMF0505S. Any compatible **insulated** DC/DC converter does the job, but have a good look at the pinout which may differ between devices.

Concerning the two crystals on the board, you may notice that the soldering pads are larger than the insulating rings of most of crystals, which can lead to unwanted short-circuits. We recommend mounting the crystals either 1 mm above the PCB, or to put a small piece of plastic between crystal and PCB.

The microcontroller crystal is used with a small **optional** inductor, which serves to force the crystal into overtone mode. Always test your board for the first time without the inductor. If you find 24 MHz on the X2 pin of the processor (with a 'scope or a frequency meter), everything is okay. If you find only 12 MHz, you have to add the inductor. If the crystal is found to oscillate at the wrong frequency, the DMX output will operate correctly, but not the MIDI input or output. In case you have neither an oscilloscope nor a frequency meter, you may test the board in Mode 2 or 4 using the DMX Tester software on the project disk. If you can get some (correct) MIDI data, you can be sure that the crystal is oscillating at the right frequency.

Table 5 : DMX connectors pinout

	Female connector	Male connector	Female connector
Pin	DMX OUT	DMX IN	DMX OUT/IN
1	Ground (SHLD)	Ground (SHLD)	Ground (SHLD)
2	Transmit DMX- (TD-)	Receive DMX- (RD-)	Transmit DMX- (TD–)
3	Transmit DMX+ (TD+)	Receive DMX+ (RD+)	Transmit DMX+ (TD+)
4	Not connected	Not connected	Receive DMX- (RD–)
5	Not connected	Not connected	Receive DMX+ (RD+)



COMPONENTS LIST

Resistors:
R1 = 470kΩ
R2= 1MΩ
R3= 1kΩ5
R5,R14,R15= 1kΩ
R4,R6-R12= 220Ω 0.5W
R13,R16= 390Ω 0.5W
R17,R18= 4kΩ7

Capacitors:
C1 = 1µF 10V
C2,C3 = 33pF
C4-C11,C17,C18,C19= 47nF
C12= 22pF
C13= 47pF
C14= 1000µF 25V radial
C15,C16= 100nF

Inductors:
L1 = 1µH5

Semiconductors:
D1,D2,D3= LED
D4= 1N4148
IC1= DS80C320MCG (Dallas Semiconductor)
IC2= 74HCT573 or 74F573
IC3= 27C256 (programmed, order code **010003-21**)
IC4= 6264 (RAM)
IC5= LTC490 (Linear Technology)
IC6= TL16C450 (Texas Instruments)
IC7,IC11,IC12= 6N137 (
IC8= 74LS04
IC9= 7805
IC10= 74LS14
IC13= NMF0505S

Miscellaneous:
B1 = B80C250
K1,K2,K3= 5-way DIN socket, PCB mount, pins at 180 degrees
PC1- PC7= solder pin
S1= 8-way DIP switch block
X1= 24MHz quartz crystal
X2= 4MHz quartz crystal
PCB, order code **010003-1**
Disk, project software, order code **010003-11**

The heatsink on IC9 regulator is almost optional, since the board has only modest power consumption. A small piece of aluminium of a few square centimetres is sufficient. If you choose a metal box (as we recommend you should), you may use one of the sides as a heatsink. Do not forget that the case of IC9 is connected to ground. It is better to isolate it from the box to avoid unexpected ground loops.

The power supply can be either a small external mains adapter (in this case, you can omit rectifier B1 and capacitor C14), or a 9 V, 5 VA mains transformer, with adequate fuse protection.

The link to DMX world is by way of a standard 5-pin XLR connector, as prescribed by the DMX specification. Since there are a lot of different pin configurations for PCB-mount XLR connectors, we decided to use chassis-mounted versions with wires to and from PCB.

It may be difficult to find male XLR 5-pin connectors for chassis mounting. Fortunately, you may avail yourself of an unclear area of the DMX specification: since pins 4 and 5 are reserved for 'optional DMX data', you may use them for DMX input. The DMX pin-out, according to the DMX specification is given in **Table 5**.

Concerning the housing, we highly recommend the use of a **metal box with a protective earth connection**. In fact, this circuit is made for professional use in 'harsh' RFI environments. No greater embarrassment than the lights system 'hanging' during a show, only because the processor is locked up as a result of RFI.

The simplest solution is to integrate the transformer inside the box, and to use a standard mains appliance socket, with an earth connection linked to box. If you want to make a 'high-end' version, you can even put a filter between the mains socket and the transformer.

One of our prototypes, we should not forget to add, was capricious when starting, giving erroneous UART operation. The only solution to get it to work properly was to add a push-button in parallel with capacitor C1 so we were able to reset the board manually, if necessary. Since fitting this button, the prototype has always worked correctly. Even if this phenomenon appeared only on one of our prototype, we should mention this workaround. In the end we discovered that the BAUDOUT pin of IC6 was stuck at 66 Hz, when it should go to 4 MHz.

With the board fully populated, it is ready for its final test.

The critical moment: testing

To test this interface, you need a DMX receiver, such as the one inside a DMX floodlight or projector. You can also hire a DMX tester with a DMX values display (light equipment rental companies will be a able to help you).

For MIDI, the easiest thing is to use a PC with a MIDI interface and running Windows 95/98, the two purpose-designed software tools will prove useful.

Connect the PC's MIDI OUT to the MIDI IN connector of interface, and connect the DMX OUT link of the interface to the DMX IN input of the floodlight (or the DMX tester). **Do not forget the DMX terminator**, especially if the DMX line is long. Configure the DMX slave (light or tester) on any DMX address, and select Mode 1 or 3 on the interface. Also select a MIDI channel and the MIDI message type on it.

Power up the interface. The POWER LED should be the only one to light up. If not, disconnect the interface and find the problem.

Launch a program named DMX TESTER, which is on the project disk. Select the MIDI interface used to connect our interface, the selected MIDI channel and the mode (1 or 3). With the 'Start' combo box, select a DMX channel corresponding to the floodlight or projector, then move the slider. The DMX slave should respond to DMX commands. If you want, you can select the 'Group' option, to send simultaneously the DMX level given by the slider on all DMX channels between 'Start' and 'Stop'.

To test the other direction (DMX/MIDI), you should have a DMX generator, like a stage lighting console. In a small number of cases you may be able to avoid it, but the test will only be on the MIDI section (note that if the first test was okay, there is little chance that the DMX section is not working properly for that direction).

Set Mode 2 (or 4) on the interface and select a MIDI channel with the associated MIDI message to send. Connect the MIDI OUT connector of the interface to the MIDI IN input on the PC. Connect the DMX generator to DMX IN on interface, if possible with a DMX terminator.

Start the DMX VIEWER program (on disk), then give it the MIDI channel and the MIDI message selected on the board. Do not forget to select a MIDI interface if necessary. The moment the interface is powered, the MIDI OUT LED of the interface should light. If you have a DMX generator, you can change DMX values and verify their evolution on the screen. If you do not, this test should be run with Delta Mode de-activated (S1-6 OFF), else the absence of changing values on DMX will cause the MIDI link to remain 'stalled'.

If, during the tests, you notice anomalies in DMX values received or transmitted (using VIEWER, you'll notice that certain DMX addresses are blocked to erroneous values), you need to consider replacing the 74F573 by a 74HCT573. The reasomn is that some RAM ICs do not mix very well with 74F ICs. In these rare cases, changing to HCT solves the problem.

How to use the interface?

This interface is suitable for many applications, including, of course, lightshows but also servicing, maintenance and lights testing.

In this last case, the easiest solution is to use DMX TESTER and DMX VIEWER on the disk (order code **010003-11**).

For lightshow use, the basic tool associated with the interface will be a MIDI sequencer. This interface is in principle compatible with all existing sequencers on PC, Mac or standalone machines. The only difference with the established use of a sequencer is that note velocities (or control values) do not control synthesizer sounds, but light intensity levels, mirror positions, gobos types and speed, etc., depending on the type of DMX device.

To control something on DMX, just record notes (or control values), with velocities (or values) corresponding to the desired DMX levels, then play the sequence and see how the DMX units behave. For the first experiments, we recommend not using the Note Off recognition or long note durations, to ensure that the DMX control has enough time to send the message. As long you 'feel' effects of MIDI to DMX, you can adapt the programming of the sequencer to your needs.

Possibilities given by a sequencer are almost unlimited, although it must be saisd that you have to be thoroughly familiar with your sequencer software or machine to rapidly arrive at a good result.

Our 'field' experience proved beyond doubt that programming and using a sequencer was a too heavy a job for a number of small and medium-size club installations, where the DJ typically doubles as the LJ (Light Jockey). For these cases, the author has developed two programs, called SoftController I and SoftController II, which allow easy and quasi-automatic control of a DMX system.

Describing these programs here would take us too far away from the goal of this article. You should know however that these software utilities represent a stage light control panel, whose look is something of a standard in the world of stage lighting. We have deleted almost all menus. To arrive at almost intuitive operation, you have to click on the buttons

On project disk 010003-11

copyright.txtl	author's copyright statements
dmx_tester.exe	DMX interface test program for Windows
midi_dll.dll	.dll file
midi_dmx.asm	EPROM contents (assembler file)
Midi_dmx.hex	EPROM contents (hexadecimal file)
mode1.asm	assembly code file for Mode 1
mode2.asm	assembly code file for Mode 2
mode3.asm	assembly code file for Mode 3
mode4.asm	assembly code file for Mode 4
viewer.exe	DMX interface viewer program for Windows

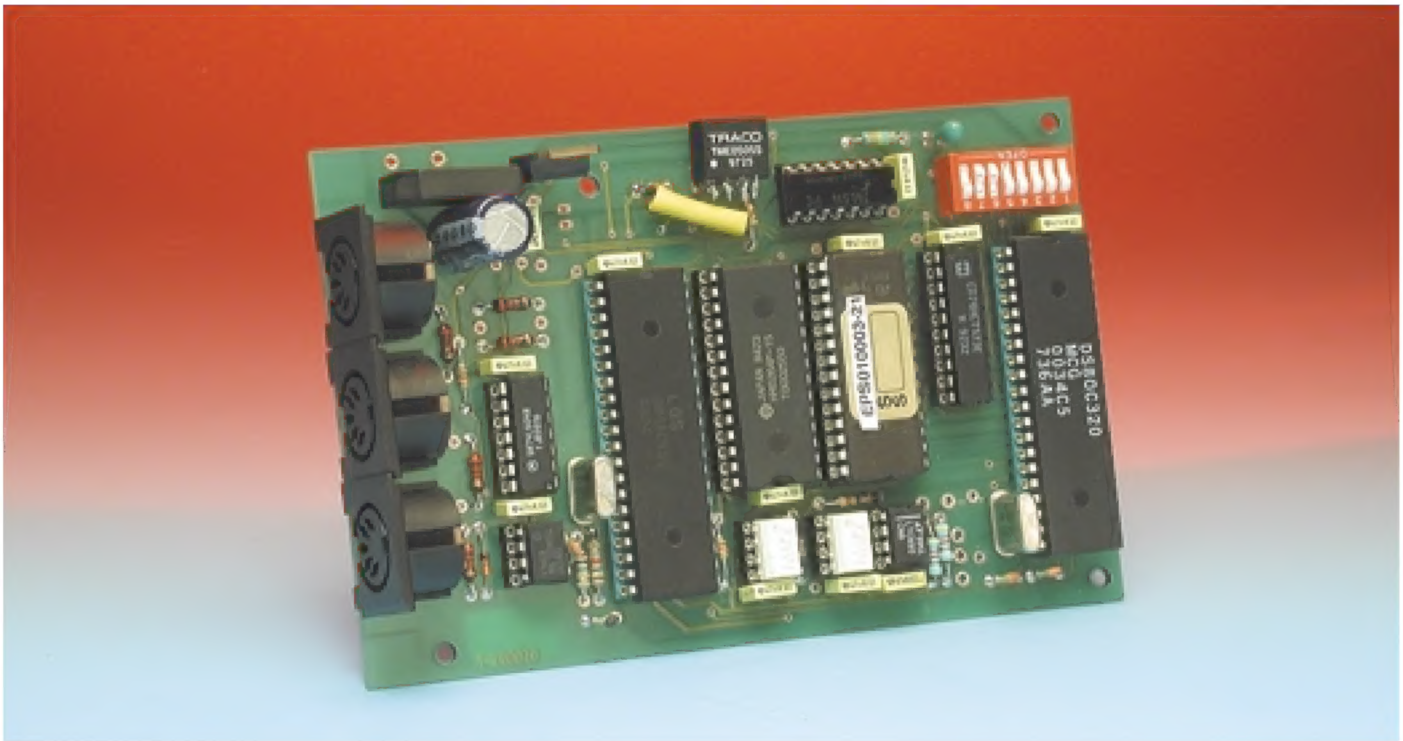


Figure 3. An early prototype of the interface board. Some improvisation was necessary around the 0505S due to a small design error which was corrected later.

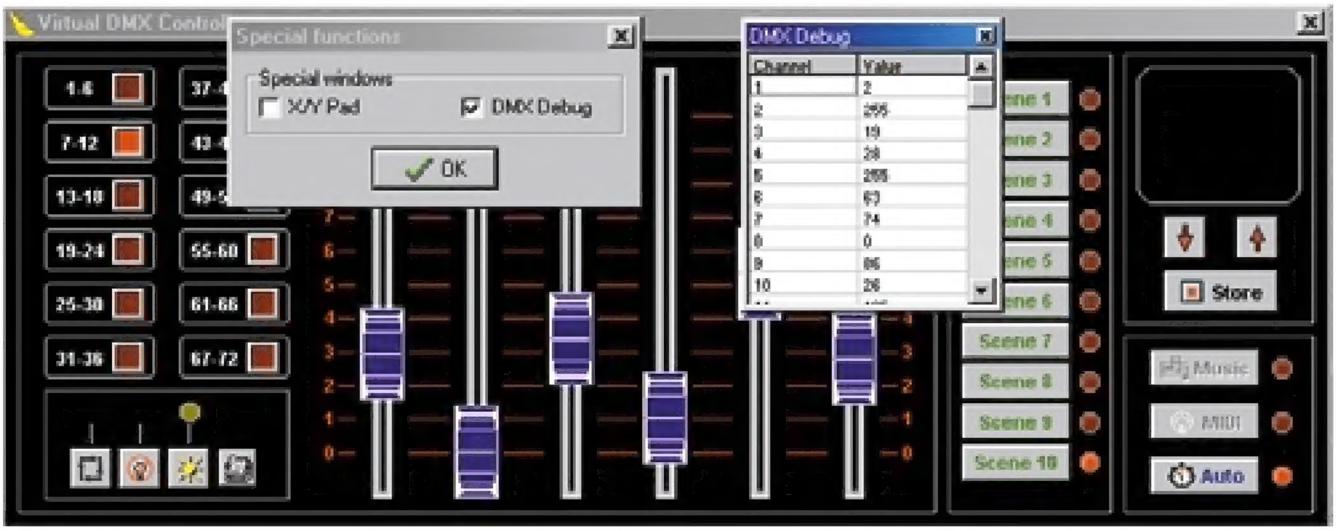
to activate them. Similarly, sliders are controlled by mouse motion, etcetera.

The software is actually employed in Belgium, to control professional lighting systems.

Finally, a bit of advice: do read the user manual (after printing it for example), before attempting to use the programs, since there are a lot of possibilities.

The software on the diskette has taken many hours of intensive work to develop and test. There is no copy protection, or evaluation version to pay afterwards, since that would only disturb 'honest' users. In spite of this, the software utilities related to the MIDI-to-DMX Interface is not 'copy-free' — by buying the project disk, you obtain only one user licence for all the software. Please do not 'give away' the software or use copies, but buy a corresponding number for your interfaces. In view of the above, the project software is not available as a free download from the *Elektor Electronics* website.

(010003-1)



Special windows

☐ X/Y Pad ☑ DMX Debug

Channel	Value
1	2
2	255
3	19
4	28
5	255
6	63
7	74
8	0
9	86
10	28

Figure 4. Scvreendump showing the SoftController program in action.